Aplicaciones RESTful (4ra parte):
Cliente Http de prueba

Teoría

Introducción

Algo vital a la hora de desarrollar un servidor es poder probar que los puntos de entrada de nuestra API funcionan como lo deseamos. Si bien nuestros servicios serán consumidos desde alguna aplicación cliente, no es necesario que utilicemos la misma para probar nuestro servidor. En este apunte aprenderemos una forma simple y fácil de realizar consultas al servidor a través de nodejs.

Módulo Request (Request-Promise-Native)

Una vez más, en lugar de utilizar las funcionalidades nativas de NodeJS para el envío de peticiones, usaremos un módulo externo, que nos abrirá una sintaxis más concisa y clara.
En este caso, usaremos el módulo 'request', y en particular, la versión personalizada que nos habilita el uso de promises, y con ello, el uso del mecanismo 'async / await' para reproducir la estructura de código sincrónico en nuestros programas.

Instalación desde la consola

```
$ npm install --save request
$ npm install --save request-promise-native
```

Uso del módulo

Para poder usar el módulo, lo primero que debemos hacer es importarlo al comienzo de nuestro archivo. El objeto obtenido luego del *require* es una función que al ejecutarla realiza la petición que deseamos, de acuerdo a lo especificado en un objeto de configuración que le pasaremos como argumento.

Ejemplo de inicialización

```
const request = require('request-promise-native')
```

Configuraciones del módulo

El módulo permite realizar todo tipo de peticiones, siguiendo el siguiente formato:

```
const options = {
    // opciones de configuración de la petición
}

try {
    const body = await request(options)
    // hacer algo!
} catch (err) {
    console.log(err)
}
```

Algunas de las propiedades del objeto de configuración

method

De tipo string, permite definir el tipo de petición que se realizará, por ejemplo 'GET', ‘POST’, ‘PUT’, ‘DELETE’, entre otros. Si no se incluye entre las opciones, toma su valor por defecto (‘GET’).

uri

De tipo string, es la ruta que identifica al recurso al que se quiere acceder. Por ejemplo:

```
http://miservidor.com/api/mensajes/1
```

json

De tipo booleano, indica que el contenido a enviarse es un objeto y debe ser transformado a json (stringify) antes de ser enviado. Su valor por defecto es *false*.

body

De tipo objeto, permite adjuntar un objeto al cuerpo de la petición. El servidor luego podrá acceder a este dato a través del campo body de la petición recibida.

qs

De tipo objeto, permite enviar parámetros adicionales, los que comúnmente se conoce como la ‘**q**uery **s**tring’ en una URL. Los campos del objeto recibido equivalen al parseo de los parámetros enviados en una URL. Ejemplo, para enviar:

```
http://miservidor.com/api/mensajes?largo=10&tema=perros
```

Se escribirá como:

```
qs: { largo: 10, tema: 'perros' }
```

resolveWithFullResponse

De tipo booleano. Por defecto, la función devuelve únicamente el cuerpo (body) de la respuesta. En ocasiones, es necesario acceder a otros detalles de la misma, además del contenido del mensaje. Para ello, basta con indicar este campo en *true*. De esta manera, la función devolverá toda la respuesta del servidor, en lugar de solamente el cuerpo de la misma. Como se dijo antes, el valor por defecto de esta opción es *false* (deshabilitado).